International Journal of Electrical and Computer Engineering (IJECE)
Vol. 6, No. 3, June 2016, pp. 1205 ~ 1212
ISSN: 2088-8708, DOI: 10.11591/ijece.v6i3.9457

Delay Comparison for 16x16 Vedic Multiplier Using RCA and CLA

M. Bhavani, M. Siva Kumar, K. Srinivas Rao
Department of Electronics and Commuunication Engineering, KLUniversity, India

Article Info

Article history:

Received Jan 6, 2016
Revised Mar 2, 2016
Accepted Mar 20, 2016

Keyword:

Carry look ahead adder
Ripple carry adder
Vedic multiplier
Verilog

ABSTRACT

In any integrated chip compulsory adders are required because first they are fast and second are the less power consumption and delay. And at the same time multiplication process is also used in various applications. So as the speed of multiplier increases then the speed of processor also increases. And hence we are proposing the Vedic multiplier using these adders. Vedic multiplier is an ancient mathematics which uses mainly 16 sutras for its operation. In this project we are using "urdhva triyagbhyam" sutra to do our process. This paper proposes the Vedic multiplier using the adders ripple carry adder (RCA) and carry look a head adder (CLA) and puts forward that CLA is better than RCA. The major parameters we are simulating here are number of slices and delay. The code is written by using Verilog and is implemented using Xilinx ISE Design Suite.

Corresponding Author:

K. Srinivas Rao,
Departement of Electronics and Communication Engineering,
KL University,
India
Email: drksrao@kluniversity.in

1. INTRODUCTION

Vedic mathematics [1] is an ancient mathematics which is mainly used by the Aryans to perform mathematical calculations. It consists of some sutras that can perform large arithmetic operation to simple calculations. After research on 8 years the Vedic mathematics was renovated by the great Indian mathematician Jagadguru Swami Sri Bharati Tirtha Maharaja. According to his research, this multiplier technique consists of mainly 16 Vedic sutras that are needed to reduce the calculation easily [2].

The Sutras along with their brief meanings are listed in [2],[3]. Of all the sutras, in this paper we are mainly using Urdhva Tiryakbhyam Sutra.

This field is very interesting and puts forward some effective algorithms which are very much utilized in various branches of engineering such as digital signal processing and computing applications. By using the ancient Vedic mathematic sutras, mainly we are using Urdhva Triyakbhyam sutra in this paper to present a simple digital multiplier architecture in which we are using two different adders like ripple carry adder and carry look ahead adder. In this paper we conclude that Vedic multiplier with carry look a head adder is faster than the multiplier with ripple carry adder and proposed adder shows it is better than carry look a head adder.

2. URDHVA TIRYAKBHYAM SUTRA

Urdhva Tiryakbhyam (Vertical & Crosswise) algorithm can be derived for "n" number of bits. The main advantage of this multiplier is, when compared with the other multipliers, the gate delay and area increases slowly as the number of bits increases [4],[5]. Therefore it is well known for time, space and power efficient multiplier. It is mentioned clearly that this multiplier architecture is fully efficient in terms of area

and speed. Since in this multiplier the partial products and their sums are calculated in parallel, the multiplier is independent of the processor's clock frequency. Therefore this multiplier is independent of the clock frequency because it will require the same amount of time to calculate the product. By choosing this vedic multiplier's structure it can be easily layout in microprocessors and designers can easily avoid this power of multiplier. It has quite regular problems to avoid tragic device failures so it can be easily increased by increasing the input and output data bus widths. The advantage is that it reduces the need of microprocessors to operate at increasingly high clock frequencies. While at this higher clock frequency generally results in increasing power and its disadvantage is that it also increases power dissipation which results in higher device operating temperatures [2].

2.1. Multiplication of two decimal numbers- 123*456

To illustrate this multiplication, let us consider the multiplication of two decimal numbers (123 * 456) shown in Figure 1 and line diagram for the multiplication is shown in Figure 2. The digits on the both sides of the line are multiplied and added with the carry from the previous step. This generates one of the bits of the result and a carry. This carry is added in the next step and hence the process goes on. If more than one line are there in one step, all the results are added to the previous carry. In each step, least significant bit acts as the result bit and all other bits act as carry for the next step. Initially the carry is taken to be zero. To make this method more clear, an alternate illustration is given with the help of line diagrams in Figure 2 where the dots represent bit 0 or 1 [2].

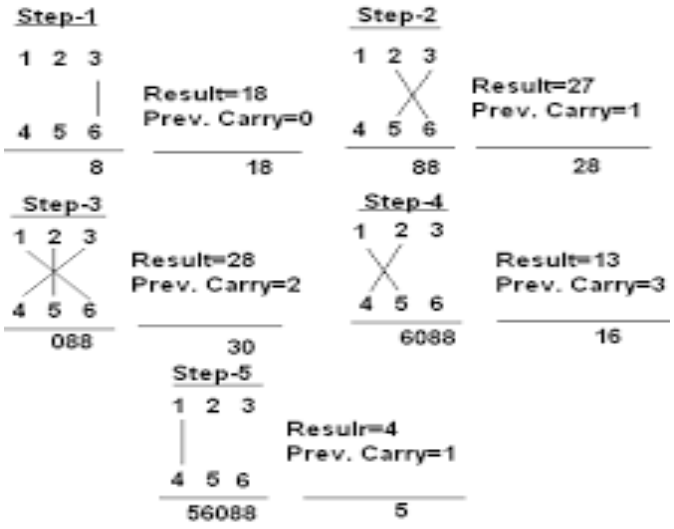

Figure 1. Multiplication of two decimal numbers

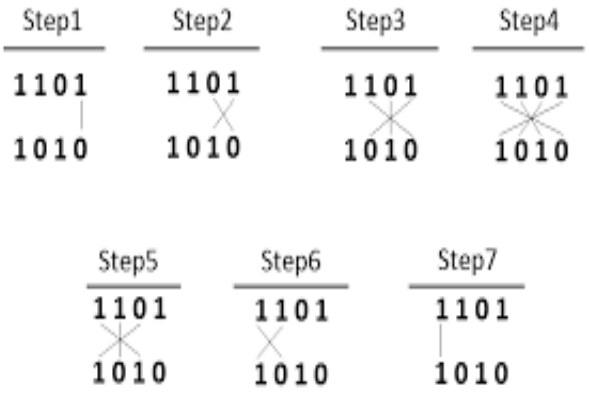

Figure 2. Line diagram of two 4 bit numbers

3. ADDER STRUCTURE

To implement this Vedic multiplier we use two different adders i.e. Ripple Carry Adder and Carry look Ahead Adder.

3.1. Ripple Carry Adder

A ripple carry adder is a logic circuit in which the carry-out of each full adder is given as the carry in of the next significant full adder which shown in Figure 3 [2]. It is called as a ripple carry adder because in the next stage each carry bit will gets rippled. In a ripple carry adder the sum and carry out bits of any half adder stage is not valid until the carry of that stage occurs. The reason behind this is Propagation delay occurs inside the logic circuit.

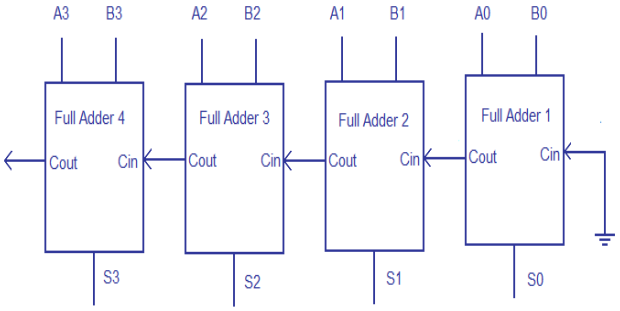

Figure 3. Ripple carry adder

The above Figure 3 represents a 4×4 ripple carry adder structure. By connecting these 4bit adders with another 4 ripple carry adders then we will get our 16×16 ripple carry adder structure. The Boolean function [6] for sum and carry for 4×4 ripple carry adder is given as follows

Sum=Ai $\bigoplus$ Bi $\bigoplus$ Ci

Carry=Ci+1 = Ai · Bi + (Ai $\bigoplus$ Bi) · Ci

3.2. Carry Look Ahead Adder:

The carry look a head adder (CLA) [2] solves the carry delay problem by calculating the carry signals in advance, based on the input signals. It is based on the fact that a carry signal will be generated in two cases: (1) when both bits Ai and Bi are 1, or (2) when one of the two bits is 1 and the carry-in is 1

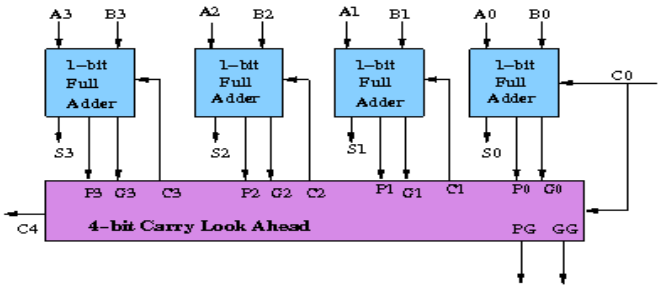

Figure 4. Carry Look Ahead Adder

Figure 4 [2] represents a 4×4 bit Carry Look Ahead Adder. By connecting these 4 bit adder for 4 times then our 16×16 carry look ahead adder will be obtained. Given the two Boolean functions [6] for the sum and carry for 4×4 CLA as follows:

SUM = Ai $\bigoplus$ Bi $\bigoplus$ Ci

Cout = Ci+1 = Ai · Bi + (A $\bigoplus$ Bi) · Ci

If we let:

Gi = Ai · Bi -- The Generate Function

Pi = (A $\bigoplus$ Bi) -- The propagate Function

Then

$C_{i+1} = G_i + P_i \cdot C_i$ -- The Carry Function

Thus, for 4-bit adder, we can extend the carry, as shown below:

$C_1 = G_0 + P_0 \cdot C_0$

$C_2 = G_1 + P_1 \cdot C_1 = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_0$

$C_3 = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0$

$C_4 = G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_0$

4. PROPOSED LOGIC

4.1. Kogge-Stoneadder

Kogge–Stone adder is a parallel prefix form carry look-ahead adder. Other parallel prefix adders include the Brent-Kung adder, the Hans Carlson adder, and the fastest known variation, the Lynch-Swartzlander Spanning Tree adder. The Kogge-Stone adder has mainly low logic depth, high node count, and minimal fan out. While a high node count implies a larger area, the low logic depth and minimal fan-out allow faster performance [6]. There are mainly three computational stages in Kogge-Stone adder. They are given below

1.Preprocessing
2.Carry generation network
3.Postprocessing

4.2. Preprocessing Stage

Preprocessing is the first stage where the generate and propagate signals of all the input pairs of signals A and B are generated separately for each bit. The logical equations of the propagate and generate signals are given by the following equations

Pi=AixorBi
Gi=AiandBi

4.3. Carry Generation Stage

Carry generation is the second stage of the KSA. At this stage the carries of all the bits are generated separately for each bit. They are divided into smaller pieces and this overall process is carried out in parallel for all the bits. Carry generate and Carry propagate bits are used as intermediate signals and their logical equations are given as follows

CPi:j=Pi:k+1andPk:j
CGi:j =Gi:k+1 or (Pi:k+1and Gk:j)

4.4. Postprocessing

This is the final step or stage of the KSA which is common for all types of adders, i.e. calculation of summation of the bits given by the logical Equations given as below

Ci–1 = (Pi and Cin) or Gi
Si= Pix or Ci–1

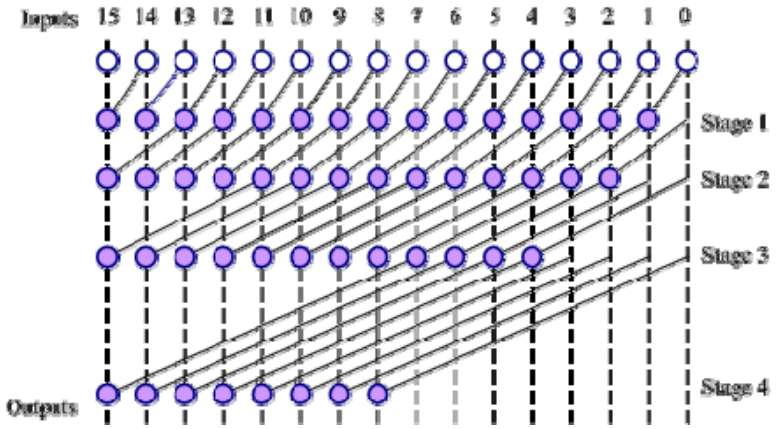

Figure 5. 16×16 Kogge stone Adder

Figure 5 represents a 16×16 Kogge stone adder with 4 stages. The stage-1 shows preprocessing, stage-2 shows carry generation and stage-3 shows postprocessing and finally fourth stage is the output representation stage.

5. RESULTS

The 16x16 Multiplier is made by using 4, 8x8 multiplier sub blocks. Here, the multiplicands are having the bit size of (n=16) whereas, the result is of 16 bit in size. The input is broken in to smaller groups of size of n/2 = 8, for both inputs, that is a and b. These newly formed groups of 2 bits are given as input to 8×8 multiplier block and the result produced 16 bits, which are the output produced from 8×8 multiplier block are sent for addition to an addition tree which is shown as hardware realization of 16 bit multiplier in Figure 6.

As the generic adder is designed the designing of high bit multipliers is not an issue using the structural modeling it becomes easy for just call the predefined components and design the multiplier.

The RTL schematics for 16×16 vedic multiplier, RCA and CLA is shown in Figure 7, 8, 9 and correspondingly the RTL for the proposed 16×16 kogge stone adder is also shown in Figure 10 respectively.

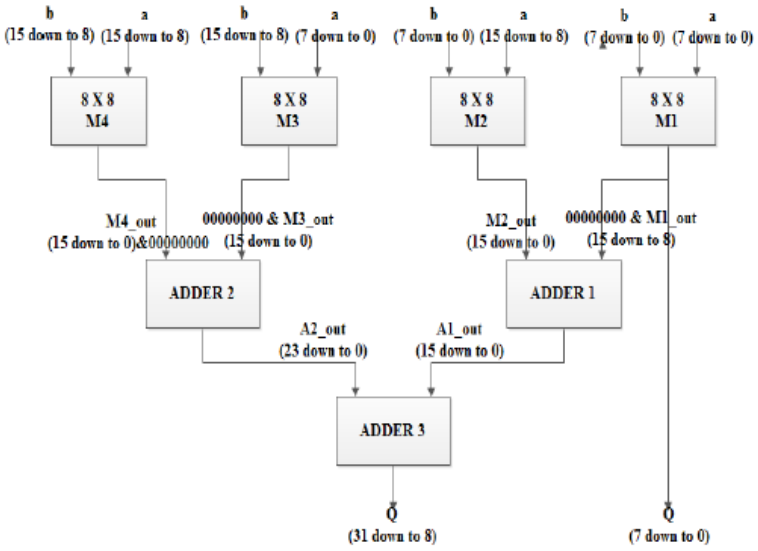

Figure 6. Hardware realization of 16×16 multiplier

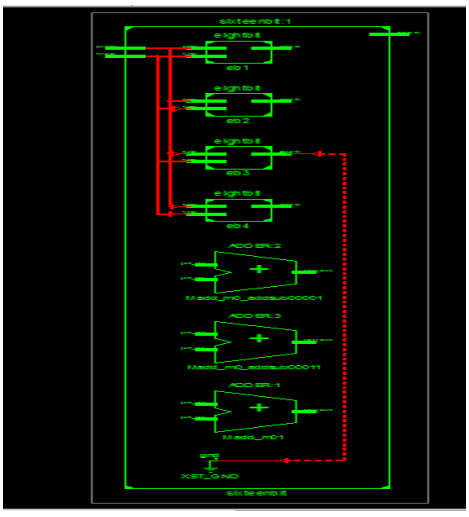

Figure 7. RTL schematic for 16×16 vedic multiplier

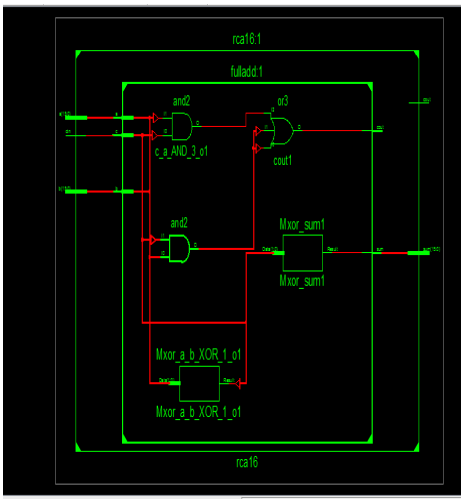

Figure 8. RTL schematic for 16×16 RCA

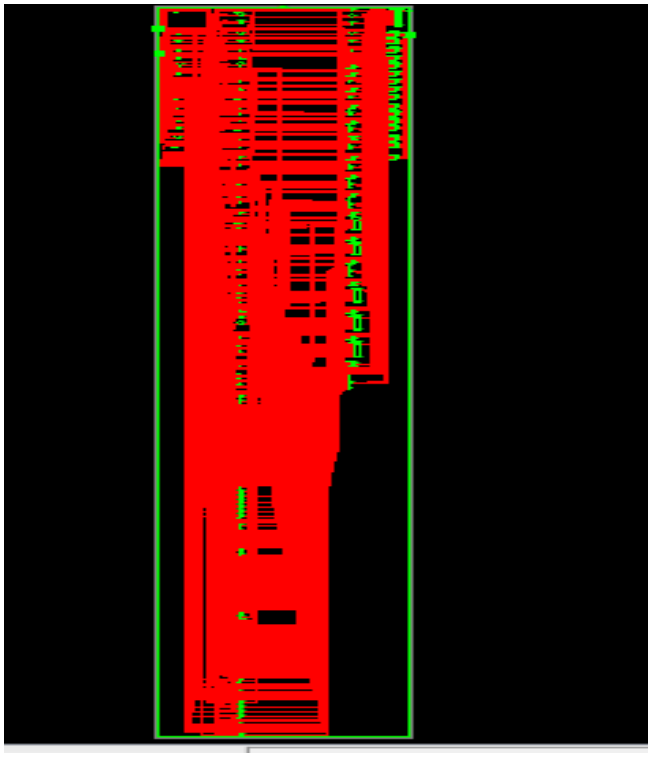

Figure 9. RTL schematic for 16×16 CLA

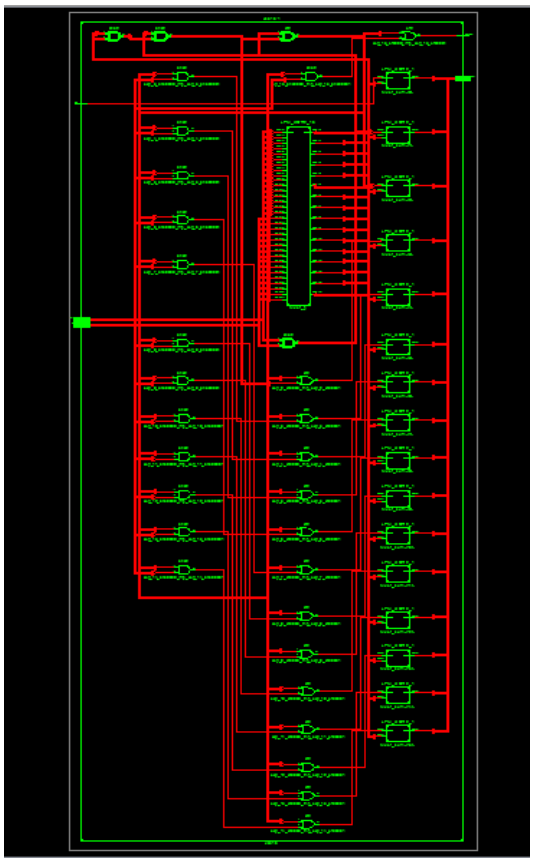

Figure 10. RTL Schematic for 16×16 kogge-stone Adder

Table 1. Comparisions Of 16×16 Vedic, RCA, CLA and proposed Kogge-Stone Adder

Parameters	Vedic Multiplier	RCA	CLA	Kogge-Stone
Delay	41.751ns	80.688ns	30.216ns	69.648ns
Slices	377	64	411	60
LUT's	387	75	716	75
Bonded IOB's	64	60	64	60

This paper presents the delay for the vedic multiplier, RCA, CLA and Kogge stone adder has the less delay when compared with the other papers [7]. Presents that the Vedic multiplier with 16 ×16 bit has a delay of 41.751ns and other parameters are given. In the same way [6] shows that the RCA, CLA and Kogge-stone adder has the delay of 80.688 ns, 78.665ns and 69.648ns respectively. But, whereas the paper [8] presents the CLA has a delay of 30.216ns and other corresponding parameters which are clearly shown in Table 1.

Finally this paper presents a less delay, slices, Look Up Tables and IOB's correspondingly when compared with the above table. The Table 2 is listed below with our values which are obtained as follows with device utilization summary also.

Table 2. Comparisions Of 16×16 Vedic, RCA, CLA and proposed Kogge-Stone Adder

Parameters	Vedic Multiplier	RCA	CLA	Kogge-Stone
Delay	25.825ns	24.686ns	21.028ns	8.955ns
Slices	350	48	25	30
LUT's	600	60	60	55
Bonded IOB's	64	50	50	65
Device Utilisation	27%	23%	23%	21%

The summary of both the adders are given separately, where it is observed that utilization of IOB's is more in the proposed adder rather than in CLA. But the amount of memory stored in form LUT's and delay are less in the proposed adder when compared with CLA.

6. CONCLUSION

This paper presents a simple and highly efficient method of multiplication mainly using the "Urdhva Tiryakbhyam Sutra" based on Vedic mathematics. It is a method for hierarchical multiplier design which clearly indicates the computational advantages offered by Vedic methods. The computational path delay for proposed 16x16 bit Kogge Stone Adder is found to be 8.955ns. By comparing with the traditional adders, its better to use the above logic for adders for any n-bit numbers, because of less complexity, and fewer number of slices, more utilization factor, less delay when compared with carry look ahead adder.

ACKNOWLEDGEMENT

I sincerely thank to my project guide, who helped me in all aspects of my project to complete in short term. We also thank KL University for providing necessary facilities towards carrying out this work.

REFERENCES

[1] J. Swami, *et al.*, "Vedic Mathematics," *Motilal Banarsidas, Varnasi, India*, pp. 40-63, 1986.
[2] G. Sharma, "Delay Comparison of 4 by 4 Vedic Multiplier based on Different Adder Architectures using VHDL," *International Journal of IT, Engineering and Applied Sciences Research,* vol/issue: 2(6), 2013.
[3] D. J. Udhani, "Implementation of High Speed Multiplier on FPGA," *International Journal of Science, Engineering and Technology Research*, vol/issue: 3(2), 2014.
[4] A. W. R. Ahmed, "FPGA Implementation of Vedic Multiplier Using VHDL," *International Journal of Emerging Technology and Advanced Engineering*, vol/issue: 4(2), 2014.
[5] A. Chouhan and A. P. Singh, "Implementation of an Efficient Multiplier based on Vedic Mathematics Using High speed adder," *International Journal of Innovative Science, Engineering & Technology*, vol/issue: 1(6), 2014.
[6] N. G. Nirmal, "Novel Delay Efficient Approach for Vedic Multiplier with Generic Adder Module," *International Journal of Engineering Research and Applications*, vol/issue: 3(3), pp. 1394-1396, 2013.
[7] M. Pradhan, *et al.*, "Speed Comparison of 16x16 Vedic Multipliers," *International Journal of Computer Applications*, vol/issue: 21(6), 2011.
[8] H. Goyal and S. Akhter, "VHDL Implementation of Fast Multiplier based on Vedic Mathematic using Modified Square Root Carry Select Adder," *International Journal of Computer Applications,* vol/issue: 127(2), 2015.